Review of Available Programs for Seasonal Adjustment of Weekly Data

Thomas D. Evans, Brian C. Monsell, Michael Sverchkov[1]
Bureau of Labor Statistics, 2 Massachusetts Ave. NE, Washington, DC 20212

Abstract
There has been a rise in interest for the seasonal adjustment of weekly data over the last few years. However, standard seasonal adjustment programs, such as X-13ARIMA-SEATS, assume constant periodicity, but weekly data can have either 52 or 53 weeks in a year. Weekly data are also difficult to seasonally adjust for multiple reasons. The week in which official holidays occur varies from year to year; some holiday effects do not occur every year; and even if the number of weeks in a year were a constant integer, the seasonal patterns would still change from year to year since the structure of the days in a month shift. The Bureau of Labor Statistics currently adjusts two weekly unemployment insurance claims series using a regression approach developed in the early 1990s, but other innovative programs are now in various stages of development. This paper will discuss various attributes of the programs, including ease of use, available diagnostics, and any technical support.

Key Words: signal extraction, high-frequency time series, calendar effects, unobserved component models

1. Introduction

Only a few papers have ventured into the realm of seasonal adjustment for weekly time series data. One of the reasons for this is that modeling weekly series is generally harder than for monthly and quarterly series. While there are well-developed and well-supported free software programs for seasonally adjusting monthly and quarterly data (e.g., see Census Bureau, 2020, and Grudkowska, S., 2017), weekly programs tend to be more complicated to use and generally less developed.

Few government agencies publish seasonally adjusted weekly time series despite their timely periodicity. The Employment and Training Administration in the U.S. Department of Labor have seasonally adjusted weekly Unemployment insurance (UI) claims series that go back to 1967. The Federal Reserve seasonally adjusts weekly raw steel production but recently stopped the adjustment of the weekly money supply series. Other agencies, such as the U.S. Census Bureau, the UK Office of National Statistics, the Australian Bureau of Statistics, and Statistics New Zealand, are all working on or interested in producing weekly adjustments.

[1] Views expressed are those of the authors and do not necessarily reflect the views or policies of the U.S. Bureau of Labor Statistics.

Weekly data do not have constant periodicity since there can be either 52 or 53 weeks in a year. A simple regression model for a differenced series *y* can consist of a seasonal component plus error:

$$\nabla Y_t = S_t\beta + e_t$$

The seasonal component can be specified as:

$$S_t = \sum_{i=1}^{k}\left(a_i \sin\frac{2\pi i Y(t)}{N_y} + b_i \cos\frac{2\pi i Y(t)}{N_y}\right) + \sum_{i=1}^{l}\left(c_i \sin\frac{2\pi i M(t)}{N_m} + d_i \cos\frac{2\pi i M(t)}{N_m}\right) \quad (1.1)$$

where N_y and N_m are the number of days in the year and the current month, Y_t and M_t are the day of the year and day of the month where week *t* ends, and *l* and *k* are the number of terms. For day-of-year effects, N_y will be equal to 365 or 366 while day-of- month effects (N_m) are 28, 29, 30, or 31. Note that if monthly effects are the same for all months, adding the within-month effects will allow fewer parameters. Another way to handle leap years could be to set the periodic effect for February 29th the same as that for February 28th. Otherwise, the leap-year effect is spread throughout the year. Whether the within-month terms are needed depends on the series. If the monthly effects are the same for all months, adding within-month effects will allow using fewer sine and cosine pairs to model the yearly effects.

Another way to model weekly seasonality is with splines. Harvey, Koopman, and Riani (1997) describes this in detail. An argument for using splines is that it can be easier to employ a more parsimonious seasonal component, but not all frequencies need to be modeled with trig seasonals to adequately capture the seasonal effects. Even if the number of weeks in a year were a constant integer, there is still a phase shift that will affect the seasonal pattern (see figure 1 in Pierce, Grupe, and Cleveland (1984) for a visual example). Thus, a weekly seasonal component cannot be modeled with dummy variables. Using periodic splines can be advantageous in certain cases but are more complex to implement and has other disadvantages. See Laidray, et al. (2018) for more details.

Holidays can be difficult in weekly modeling. They can be thought of as special seasonals since they move across weeks as holidays like Easter and Ramadan move across months. While the U.S. Thanksgiving holiday is always in the month of November, it can be in week 47 or week 48 (BLS uses Saturday as the reference day). Easter can be in either March or April but will also vary across weeks 12-18. Thanksgiving can be in either week 47 or 48, but if it falls "late" in the month, the effect can be different. Thus, removing the holiday effects from the seasonally adjusted series might be wise.

The first publicly available software program specifically designed for the seasonal adjustment of weekly data is probably the *CATS* (Calendar and Times Series) program (Cleveland 1986) written by William P. Cleveland based on work by Pierce, Grupe, and Cleveland (1984). The program is written in Fortran and was used by the Bureau of Labor Statistics (BLS) to seasonally adjust UI data for many years. The program has many capabilities for seasonal adjustment but only produces deterministic seasonal factors. It can also handle monthly and quarterly data. Different weighting patterns are built into the program for holidays. In addition, ARIMA models can be fit to the residuals, and forecasts can be made for the original series and the seasonal factors.

The current program used by BLS to officially seasonally adjust the UI claims series was developed by Cleveland during the early 1990s at the Federal Reserve (See Cleveland, Evans, and Scott 2018). Changes were made to the original Fortran code by BLS staff, and the program is called *MoveReg* for "moving regressions." While this program has served BLS well, we are looking to move to another program that is more flexible and that explicitly models the time series components. As interest in seasonally adjusting weekly data has increased recently, other programs are now available or in development. The purpose of this paper is to briefly introduce, review, and compare different programs.

Andrew Harvey popularized the used of structural time series (STS) models to analyze and seasonally adjust economic time series data (see Harvey 1989). A base structural model (BSM) consists of trend, seasonal, and irregular components. However, as STS models are highly flexible, a BSM model can also easily be extended to include calendar, sampling error, and outlier components as needed. STS models can also handle high-frequency data (see Harvey, Koopman, and Riani (1997) and Harvey and Koopman (1993)). It is effectively a regression model with time-varying coefficients with smoothing capabilities. Rajesh Selukar developed ways to implement STS models in Proc SSM (SAS 2020) and Proc UCM in the SAS statistical software package. Many of SSM's and UCM's algorithms are based on work by De Jong (e.g., De Jong 1989). For this paper, we choose to use SSM although UCM is also appropriate (see Selukar 2011). Note that an STS model can be extended to higher frequencies such as daily or hourly (Harvey and Koopman 1993).

Recently, there is new work on modeling seasonality with ARIMA models. Jean Palate of the National Bank of Belgium is developing an R implementation of a fractional airline model (FAM) in state-space form where the seasonal periodicity is 52.17 for weekly series with the usual canonical decomposition of SEATS. This model can also be extended to higher periodicities and is briefly explained in Section 2.2.

Other programs that could be considered for weekly seasonal adjustment are *STL* (Cleveland, et al. 1990), and the *KFAS* routine in the R software package. *STL* is a filtering procedure that is based on the loess smoother and produces trend, seasonal, and remainder components. *STL* never really caught on across statistical agencies so we do not evaluate it here. The Office of National Statistics in the UK is looking at FAM and will likely report on their experiences. *KFAS* has many similarities to Proc SSM and is likely a suitable approach. Another possibility is *Ecce Signum* (*Sigex*) which is under development by Tucker McElroy and James Livsey of the U.S. Census Bureau (McElroy and Livsey 2020). This package is available in R and can handle various multivariate models. A download is available online on GitHub.

The remaining layout of this paper is as follows. Section 2 reviews the weekly seasonal adjustment programs that are either currently available or in development: Sections 2.1-2.3 describe our three programs selected to review; Section 3.1 discusses our data; and Section 3.2 compares output from *MoveReg*, SSM, and FAM for UI data. Section 4 offers a summary.

2. Weekly Programs

2.1 *MoveReg*

The advantage of the *MoveReg* program over the earlier *CATS* program is that it uses locally-weighted regressions to allow stochastic seasonal factors. Separate regressions are used for each year with the same seasonal model but different weights. The program works well but is more difficult to set up and run than, e.g., *X-13ARIMA-SEATS*. Many changes were made to the original Fortran code by BLS staff, and the program was named as *MoveReg* for "moving regressions." A detailed description of the program and its optional SAS interface is in Cleveland, Evans, and Scott (2018). *MoveReg* has been used by BLS since 2002 to seasonally adjust the weekly UI claims series.[2] The current version is now 6.0. Twelve holidays are built into the program with choices of weighting schemes. User-supplied holidays can also be used. Outliers can be included by adding in the year and week. Running a series is very quick. Since *MoveReg* uses locally-weighted regressions for stochastic seasonal factors, the program likes data to be input in full years.

The series are always differenced in *MoveReg*, but the program has been recently updated to use either additive or multiplicative models, level shifts (traditional 0s and 1s or the X-13 style with -1s and 0s), and temporary change outliers (see Monsell 2021). Note that BLS uses trig seasonals as shown in Equation 1.1 (but without within-month effects).

With the IC data, every week from the onset of the pandemic through 2020 is an additive outlier (AO). One way to utilize parsimonious outlier effects during the Covid-19 pandemic, is to consider using all three of the usual outlier types: AOs, temporary changes (TC), and level shifts (LS). Otherwise the process will be equivalent to the projected factor seasonal adjustment method for the whole pandemic period.

The basic steps are:

1) Fit a fixed global regression without weights to estimate holiday and outlier effects
2) Remove the trend, calendar, and outlier effects to make y^*
3) Use separate regressions for y^* for each year with the same seasonal model but different weights
4) Restore trend, outlier, and holiday effects
5) Compute the projected seasonal forecast factors
6) Return values to original scale

A few settings need to be determined in advance:

1) AR coefficient
2) Variance ratio
3) Number of terms in the trig seasonals
4) Holidays and their weights
5) Outliers (AOs, LSs, and TCs)
6) TC decay factor (if TC outliers are specified)

Cleveland, Evans, and Scott (2018) assist in how to determine some of the settings. Also see Monsell (2020) for more information on program updates. As holidays and outliers are

[2] See https://www.bls.gov/lau/seasonal-adjustment-for-weekly-unemployment-insurance-claims.htm for details.

in their own components, the user can decide what to include in the final seasonally adjusted series.

Given the model

$$y_t = \mu_t + e_t$$
$$(1-B)(1-\phi B)\mu_t = a_t$$

with white noise terms e_t and a_t, the weights to estimate μ_t given y form the desired W^* matrix. These are obtained from

$$E(u \mid y) = (I + v \textstyle\sum_{\mu}^{-1})^{-1} y = W * y$$

The AR coefficient and the variance ratio affect the smoothness of the seasonal factors. For example, setting $\phi = 0.5$ and $\upsilon = 16$ is somewhere between the X-11 3x5 and 3x9 seasonal filters.

A typical additive model can be:

$$\nabla Y_t = S_t + H_t + O_t + e_t$$

where ∇Y_t is the differenced observed series (at time *t*), *S* is the seasonal component, *H* is the holiday component, *O* is the outlier component, and *e* is an error term. In practice, the series is first differenced (and sometimes logged), and the trend component is not explicitly modeled.

A few diagnostics are provided including Box-Ljung statistics, a table of ANOVA statistics, and t-values/p-values for outlier, holiday, and seasonal coefficients. The optional SAS interface adds nine plots including the spectra, revisions, sub-plots by trig coefficients, and seasonal sub-plots by week.

An example of the input control file is here:

```
52 0 0
3 17 1 13  2 2 1   nout nls ntc hol nfilt mxtype lsx13
0.92               tc decay factor
0.4  16            phi sigratio
60                 number of trig coefficients
2003 01 30 2021 01 30 7
49 2008  47 2019  28 2020
13 2020
0 0 0 3  10  8  1   4   7   2  12  11   9
8  1  0 0 0 0 0 0 1. 1.      New Year
1  1  1.                     M. L. King Birthday
1  1  1.                     Presidents Day
8  8  1.  0 0 0 0 0 0 0      Easter
1  1  1.                     Memorial Day
1  1  1.                     July 4th
2  2  0. 1.                  Labor Day
1  1  1.                     Columbus Day
1  1  1.                     Veterans Day
1  1  1.                     Thanksgiving
```

Outliers are listed in order by type (AOs first, LSs, TCs). More details are in Cleveland, Evans, and Scott (2018) and Monsell (2020).

Variances for week-to-week changes were researched in Evans and Sverchkov (2016). A parametric bootstrap is employed. The procedure is straightforward and likely to be helpful.

2.2 Fractional Airline Model

The fractional airline model is an implementation developed by the Bank of Belgium as part of an R package that accesses a Java library under development for the JDemetra+ program (see Grudkowska 2017). JDemetra+ is a tool for seasonal adjustment developed by the National Bank of Belgium in cooperation with the Deutsche Bundesbank and Eurostat in accordance with the Guidelines of the European Statistical System (ESS).

The *rjdhighfreq* package contains routines for the modelling and seasonal adjustment of high-frequency series (see Palate 2020). One of those methods is the fractional airline decomposition examined in this paper. A fractional airline model can be viewed as first-order Taylor expansion of an "airline model" (ARIMA (0 1 1) (0 1 1)s, where s is the seasonal period) with non-integer periodicity (in the weekly case, s = 365.25/7 = 52.1726). In the case of a weekly model, the autoregressive polynomial of the model is

$$(1-B)((1-0.82B^{52})-(0.18B^{53}))$$

and the moving average polynomial is

$$(1-\theta_1 B)\big((1-0.82\theta_{52}B^{52})(1-0.18\theta_{53}B^{53})\big).$$

A canonical decomposition is performed as in Burman (1980) and the different components are estimated by means of the Kalman smoother (with exact diffuse initialization). These adjustments are analogous to those of the *SEATS* software developed by Gómez and Maravall (see Gómez and Maravall 1997 and Gómez and Maravall 2001).

This version of the high dimensional modeling R package includes several useful innovations, including:

- Automatic outlier identification for level shift and point outliers, with the ability to specify a critical level for the outlier t-statistic
- Fixed regression terms, which can be used to incorporate holiday and other outlier effects
- An optional standard error term for the signal extraction components
- The ability to generate forecasts and backcasts
- R routines that can be used to generate moving holiday regressors to include in the joint model

More options and output enhancements are expected in newer versions. Currently, FAM gives you AICs, model coefficients and their T-values.

To facilitate comparison with the other methods, we used holiday regressors as defined in the *MoveReg* Fortran program. These holiday regressors were generated from R functions developed at BLS. Other R functions allow for the generation of holiday and outlier effects, FAM model summaries, and the assignment of outlier effects to different components of the signal extraction (AOs to the irregular component, LSs to the trend component), and are available in an R package developed by Brian Monsell. Some of these features may be included in future FAM updates. An example of R code for running FAM is included in the appendix.

2.3 SSM

A base structural model (BSM) consists of trend, seasonal, and irregular components. However, as STS models are highly flexible, a BSM model can also easily be extended to include calendar, sampling error, and outlier components as needed. STS models can also handle high-frequency data (see Harvey, Koopman, and Riani (1997) and Harvey and Koopman (1993)). It is effectively a regression model with time-varying coefficients with smoothing capabilities. Two ways to implement an STS model are with SSM or UCM in the SAS statistical software package. For this paper, we chose to use SSM although UCM is also appropriate.

SSM is a powerful modeling procedure and has many options for time series data. For example, SSM can be easily adapted to perform ARIMA modeling, account for sampling error, forecasting, and can model the trend component in several different ways. Also, outlier and holiday components can be specified as needed in the model to handle many different effects. Missing values are also not a problem. Six different optimizers for maximum likelihood estimation give the user choices depending on the problem. Automatic detection is available for level shifts and additive outliers. Speed is not a big problem and SSM works well in both the Linux and PC environments. Scaling can be very important with SSM. A suggestion is to scale the observed series between 1 and 100. Fixing the hyperparameters during official production can increase execution speed.

Some disadvantages for SSM are obvious for those who are used to using a seasonal adjustment program such as *X13ARIMA-SEATS*. SSM does not choose between additive and multiplicative models, automatically detect temporary change outliers, test for trading day effects, etc. It is also missing some key diagnostics for seasonal adjustment, but they can be easily programmed in SAS. A key disadvantage is that SSM does not automatically produce forward-filter estimates that many of us expect with Kalman filter programs. However, the one-step-ahead prediction errors that are needed for diagnostics can be produced by iterating through the series and this takes much less time that one might expect. Graphical diagnostics can easily be produced with SAS/Graph or Proc SGPLOT or from many other statistical packages.

The seasonal component can be structured for weekly data with trigonometric seasonals as shown in Equation 1.1, dummy variables, or time-varying periodic splines. We choose to use the same trig seasonal model here as for *MoveReg* (the default seasonal component does not account for leap years). Using periodic splines can be advantageous in certain cases but is more complex to implement and has some disadvantages. See Laidray, et al. (2018) for more details.

Holidays are handled similarly to those in *MoveReg*, but we center them. The New Year holiday's weight is split between the first two weeks as in *MoveReg*. Since they are all "moving," we delete the holiday effects from the final seasonally adjusted series.

An additive decomposition model for SSM in our case is:

$$Y_t = T_t + S_t + I_t + H_t + O_t$$

where Y_t is the observed series (at time *t*), *T* is a trend component, *S* is a seasonal component, *I* is an irregular component, *H* is a holiday component, and *O* is an outlier component. The trend component consists of a local linear trend with a level and slope to form an integrated random walk.

An example of Proc SSM code for weekly seasonal adjustment is below. The trig regressors are defined outside the proc since the periodicity is not constant. The holidays and outliers are defined in macro variables to simplify any changes.

3. Results from *MoveReg*, FAM, and SSM

3.1 Data

The data used in this paper are initial claims (IC) as collected by states for the UI program and produced by the Employment and Training Administration in the U.S. Department of Labor. The states are required to submit their data by the close of business each Monday for the number of the initial claims to receive benefits from the UI program. The number of initial claims naturally rise sharply during recessions but rose to unseen levels during the current pandemic and remains high well into 2021. The IC series has also recently been affected by extended benefits and extra benefits from the Pandemic Unemployment Assistance program.

Pandemic effects began for IC in the second week of March 2021. By April 2021, the unadjusted IC exceeded 6,000,000. Until the pandemic, the only time IC reached 1 million was during a recession in the second week of January 1982 (seasonality is always strongest in the second week). Since the beginning of the pandemic, twenty weeks have claims that exceed a million. Before 2020, IC averaged less than 400,000 per week, but averaged over 1.6 million during the pandemic in 2020. It is clear from Figure 1 as to how much effect the pandemic has on IC. As of July 2021, IC still has not returned to "normal" levels, but we expect that may happen soon due to the elimination of pandemic payments and the overall improvement of the U.S. economy.

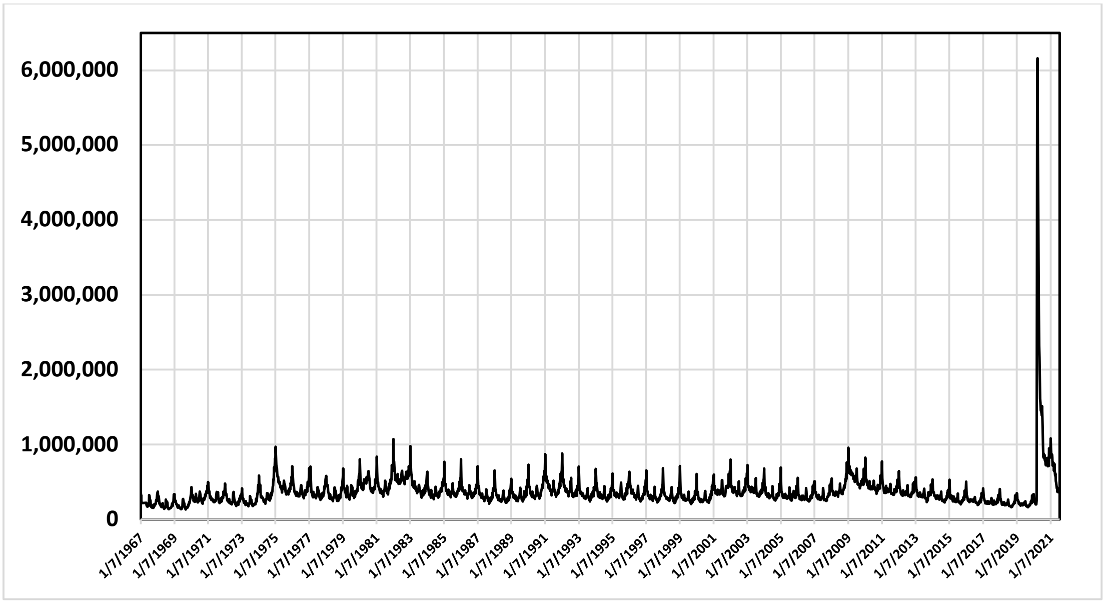

Figure 1: Historical Initial Claims, Unadjusted

Figure 2 shows the average multiplicative seasonal factors by week before the pandemic period as produced by *MoveReg*. The strongest periods for seasonality start in November and continue through early January. Seasonality is also relatively strong around the July 4th holiday.

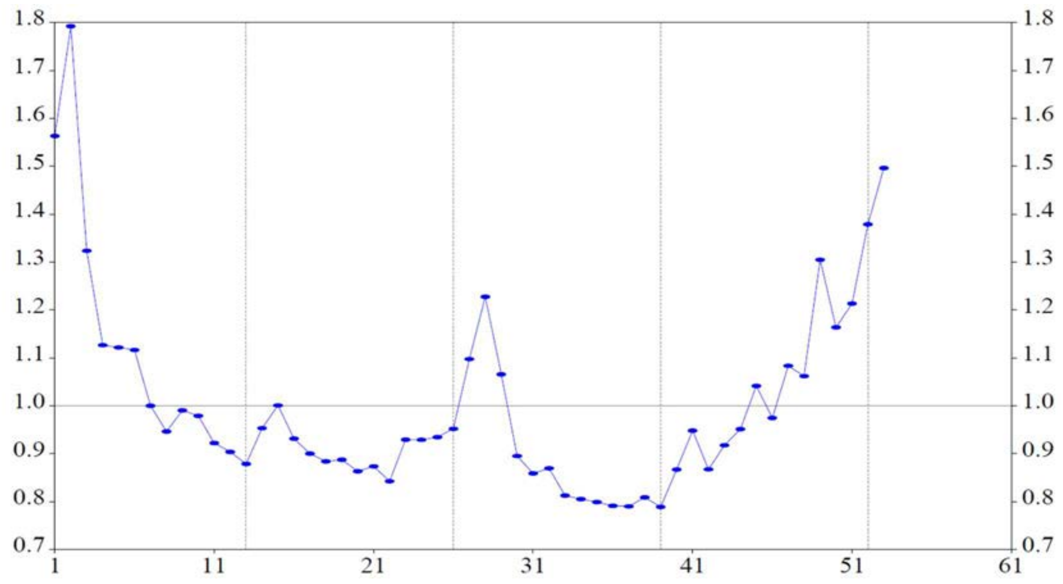

Figure 2: Average Seasonal Factors by Week of Year, 1991- 2020

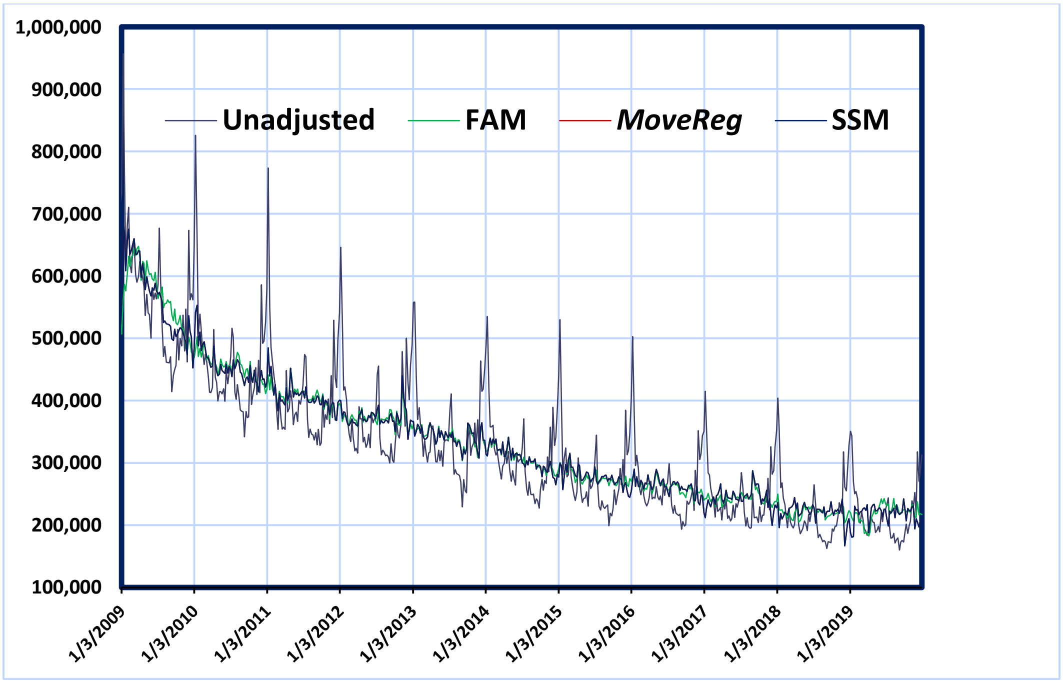

Figure 3: Initial Claims, Pre-Pandemic

3.2 Results

Figure 3 shows the seasonally adjusted series from the three programs from 2009 through 2019 with the unadjusted IC. It is hard to tell the difference between the adjusted series throughout most of the period. Some of the larger differences can be due to outlier sets. One area that might be helpful to research is to whether the holiday variables should be estimated stochastically. This is easy to do in SSM.

To see more detail, Figure 4 shows the same series for 2018. There are some differences, but the adjustments are similar overall.

The pandemic period in 2020 is shown in Figure 5. Note the abrupt rise starting in March and the rapid decline just afterwards. A TC outlier (or possibly multiple TCs) looks obvious, but we found that combinations of AOs and LSs work better. We use 0.92 for the TC decay rate with weekly data since it is equivalent to using 0.7 for monthly data. Figure 6 displays the pandemic in 2020 starting in June to be able to see differences more clearly. One noticeable difference is in the second week of July, which is treated as an AO. This is probably due to July 4th falling on a Saturday that likely caused many claims to be delayed until the next week.

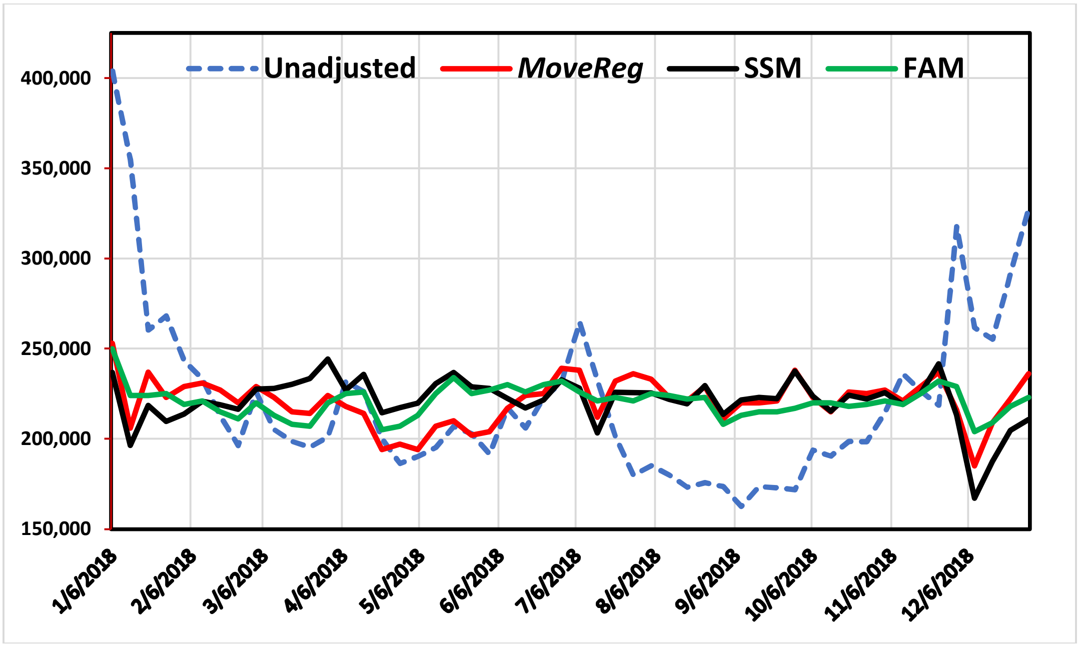

Figure 4: Initial Claims, 2018

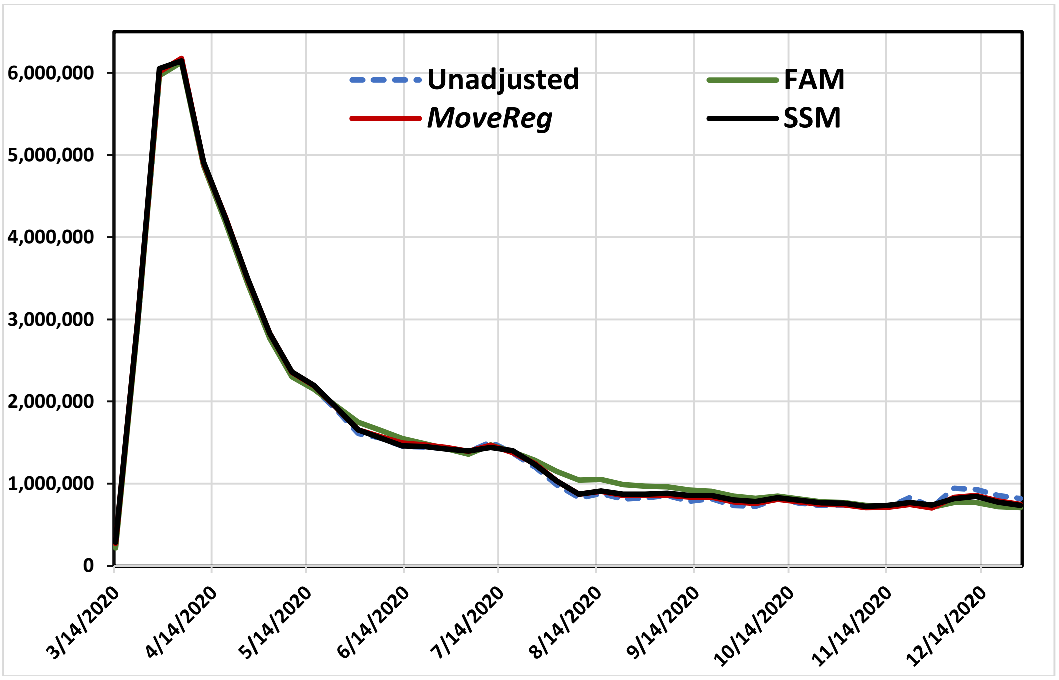

Figure 5: Initial Claims, 2020 Pandemic Period

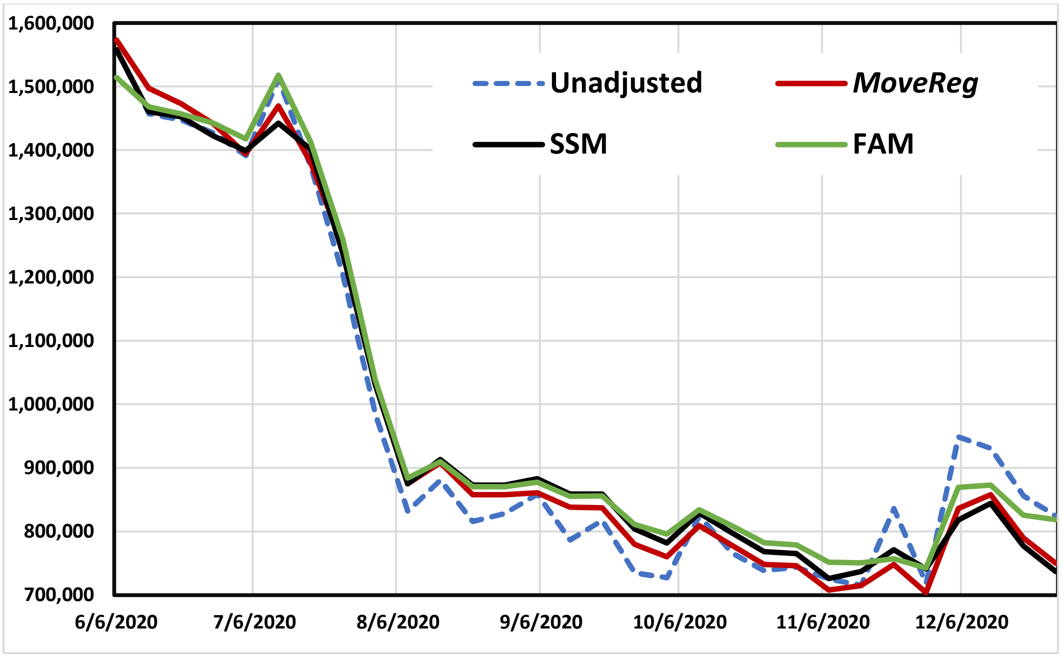

Figure 6: Initial Claims, June – December 2020 Pandemic Period

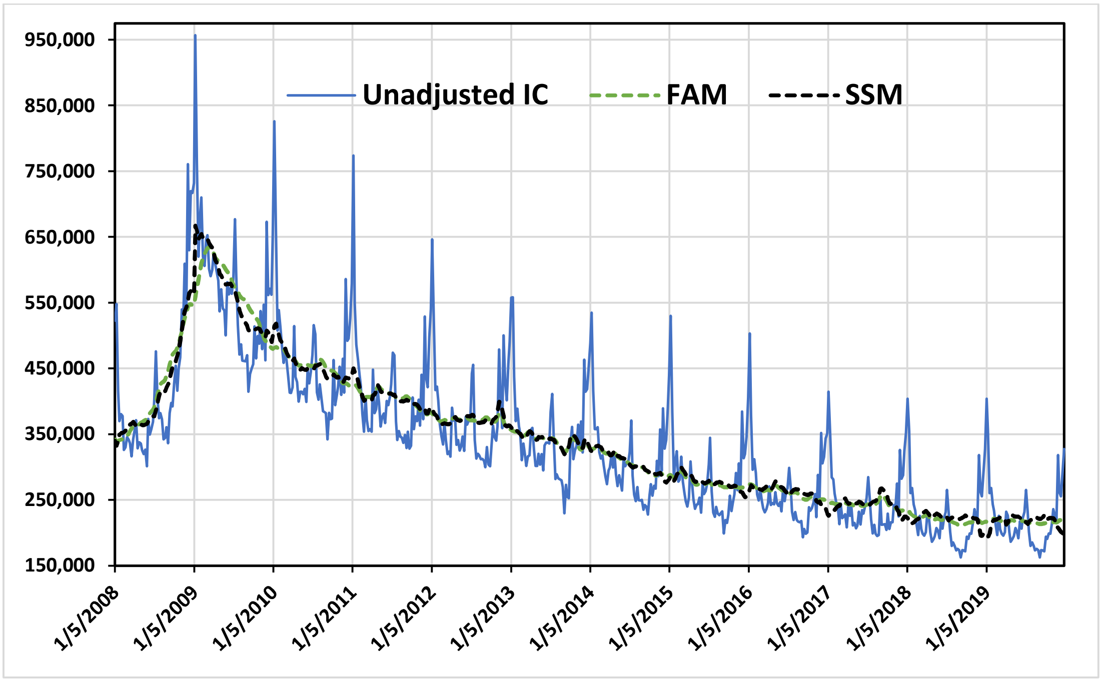

Figure 7: Initial Claims, Pre-Pandemic FAM and SSM Trends

Trends from FAM and SSM are plotted in Figure 7. Again, they are similar over most of the period. (*MoveReg* does not explicitly estimate a trend component.) It could be useful to publish trends along with the seasonally adjusted series.

4. Summary

A natural question arises now: “Which program should you use”? All three appear to work well for our weekly data series, but *MoveReg* has some shortcomings as mentioned earlier.

Below are some points to consider:

- FAM is still in development, but many will find it helpful that it is designed to perform seasonal adjustment. SSM is a powerful general time series procedure but will require the user to create some seasonal adjustment diagnostics separately.
- FAM uses an ARIMA model and SSM an STS model. Some users might feel more comfortable with one approach over another. Regardless, based on Maravall (1985), we would expect a base structural model and an airline model to be similar.
- While one might be tempted to choose between FAM and SSM based on one’s comfort level with R or SAS, being an expert in either language is not required. A solid knowledge in time series and seasonal adjustment methods is likely more important.

Acknowledgements

Thanks to Jean Palate of the National Bank of Belgium for providing us with a preliminary version of his fractional airline modelling software, and to Richard Tiller and Reid Rottach of the Bureau of Labor Statistics for their discussions and advice.

Appendix A

Example of R code for FAM:

```
setwd("X:/code/weeklyAdjustment")

ic_df <- openxlsx::read.xlsx("uihis_new.xlsx", sheet=1)
uihis_start <- c(1988, 1)

ic_tis <- tis::tis(ic_df$ic, start = uihis_start, tif =
"wsaturday")

ic_week_tis <- tis::tis(ic_df$week, start = uihis_start, tif =
"wsaturday")
ic_year_tis <- tis::tis(ic_df$year, start = uihis_start, tif =
"wsaturday")

ic_series_end <- end(ic_tis)
ic_forecast_end <- ic_series_end + 104


# set starting date to the last January in 2003
this_start <- c(2003, 4)

# set ending date to last January in 2021
this_end   <- c(2021, 5)
```

```
# create series, week and year variables with this starting and
ending date

ic_tis_final        <-
   window(ic_tis, start=this_start, end=this_end)
ic_week_tis_final   <-
   window(ic_week_tis, start=this_start, end=this_end)
ic_year_tis_final   <-
   window(ic_year_tis, start=this_start, end=this_end)

ic_this_start <- this_start

ic_obs         <- ic_tis_final
ic_length      <- length(ic_tis_final)

ic_ti <- tis::ti(ic_tis_final)
ic_end <- end(ic_tis_final)

# generate holiday, outlier regressors for fractional airline
# model using gen_movereg_holiday function from airutilities
# package developed by Brian C. Monsell

ic_ny  <-
  airutilities::gen_movereg_holiday(hol_n = 8,
                      hol_index = 1,
                      hol_wt = c(0, 0, 0, 0, 0, 0, 1, 1),
                      hol_type = "newyear",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_mlk <-
  airutilities::gen_movereg_holiday(hol_n = 1,
                      hol_index = 1,
                      hol_wt = array(1, dim=1),
                      hol_type = "mlk",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_president <-
  airutilities::gen_movereg_holiday(hol_n = 1,
                      hol_index = 1,
                      hol_wt = array(1, dim=1),
                      hol_type = "president",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_easter  <-
  airutilities::gen_movereg_holiday(hol_n = 8,
                      hol_index = 8,
                      hol_wt = c(1, 0, 0, 0, 0, 0, 0, 0),
                      hol_type = "easter",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_memorial <-
  airutilities::gen_movereg_holiday(hol_n = 1,
                      hol_index = 1,
                      hol_wt = array(1, dim=1),
                      hol_type = "memorial",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_july4 <-
  airutilities::gen_movereg_holiday(hol_n = 1,
                      hol_index = 1,
                      hol_wt = array(1, dim=1),
                      hol_type = "july4",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_labor  <-
  airutilities::gen_movereg_holiday(hol_n = 2,
                      hol_index = 2,
                      hol_wt = c(0, 1),
                      hol_type = "labor",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_columbus <-
  airutilities::gen_movereg_holiday(hol_n = 1,
                      hol_index = 1,
                      hol_wt = array(1, dim=1),
                      hol_type = "columbus",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_veteran <-
  airutilities::gen_movereg_holiday(hol_n = 1,
                      hol_index = 1,
                      hol_wt = array(1, dim=1),
                      hol_type = "veteran",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

ic_thanksgiving <-
  airutilities::gen_movereg_holiday(hol_n = 1,
                      hol_index = 1,
                      hol_wt = array(1, dim=1),
                      hol_type = "thanksgiving",
                      this_week = ic_week_tis_final,
                      this_year = ic_year_tis_final)

# Generate “special” holiday regressors using various routines
# from the airutilities package

ic_july4_wed <-
   airutilities::match_month_day(ic_week_tis_final, "0707")
ic_xmas_w53  <-
   airutilities::match_week(ic_week_tis_final, 53)
ic_xmas_fri  <-
   airutilities::match_month_day(ic_week_tis_final, "1226")

# Create holiday regression matrix by binding individual holiday
# regressors and set the column names
```

```
ic_holiday_matrix <-
  cbind(ic_ny, ic_mlk, ic_president, ic_easter, ic_memorial,
        ic_july4, ic_labor, ic_columbus, ic_veteran,
        ic_thanksgiving, ic_july4_wed, ic_xmas_w53,
        ic_xmas_fri)

colnames(ic_holiday_matrix) <-
      c("ny", "mlk", "president", "easter", "memorial", "july4",
        "labor", "columbus", "veteran", "thanksgiving",
        "july4_wed", "xmas_w53", "xmas_fri")

# construct outliers for outlier set with TC by creating date
# matrix and using gen_tc_outlier_matrix function from the
# airutilities package

ic_firstTC_date <-
  matrix(c(13, 2020), ncol=2, byrow=TRUE)

ic_firstTC_matrix <-
  airutilities::gen_tc_outlier_matrix(ic_firstTC_date,
                                      ic_week_tis_final,
                                      ic_year_tis_final, 0)

# bind holiday and TC matrix together to form final regression
# matrix, setting the column names from the original parts

ic_tc_auto_matrix <- cbind(ic_holiday_matrix, ic_firstTC_matrix)

colnames(ic_tc_auto_matrix) <-
  c(colnames(ic_holiday_matrix), colnames(ic_firstTC_matrix))

# Generate ljung-based critical value for outliers using the
# set_critical_value function from the sautilties package
# developed by Brian C. Monsell

ljung_cv <-
   sautilities::set_critical_value(length(ic_tis_final),
                                   cv_alpha = 0.005)

# fractional airline with holiday regressors,
# alternate outlier set with TC, no log, using ljung_cv
# Estimation and decomposition generated from the
# fractionalAirlineEstimation and fractionalAirlineDecomposition
# of the rjdhf package, developed by Jean Palate

ic_tc_auto_ljung_nolog_est <-
   rjdhf::fractionalAirlineEstimation(ic_tis_final,
                                      periods=c(365.25/7),
                                      x=ic_tc_auto_matrix,
                                      outliers=c("ao", "ls"),
                                      criticalValue = ljung_cv)
ic_tc_auto_ljung_nolog_decomp <-
   rjdhf::fractionalAirlineDecomposition(
      ic_tc_auto_ljung_nolog_est$model$linearized, 365.25/7,
      stde = TRUE)
```

```
# generate a matrix containing a summary of the FAM fit to the
# series with regressors with the gen_air_model_matrix function
# from the airutilities package

ic_tc_auto_ljung_nolog_model <-
  airutilities::gen_air_model_matrix(ic_tc_auto_ljung_nolog_est,
                     xreg_names = colnames(ic_tc_auto_matrix),
                     this_week = ic_week_tis_final,
                     this_year = ic_year_tis_final)

# check model and set up codes for different types of regressors:
# “ao” for point outliers, “ls” for level changes,
# “tc” for temporary change outliers, and “hol” for holiday
# regressors

this_row_names <- rownames(ic_tc_auto_ljung_nolog_model)

this_otl_index <-
  sort(c(grep("ao", tolower(substr(this_row_names, 1, 2))),
       grep("ls", tolower(substr(this_row_names, 1, 2))),
       grep("tc", tolower(substr(this_row_names, 1, 2)))))

this_xtype_otl <-
  tolower(substr(this_row_names, 1, 2))[this_otl_index]

this_xtype <-
   c(rep("hol", ncol(ic_holiday_matrix)), this_xtype_otl)

# generate a list object with the components of the fractional
# airline decomposition with outliers assigned to the components
# as they are in X-13ARIMA-SEATS with the gen_air_components
# function of the airutilities package

ic_tc_auto_ljung_nolog_comp <-
  airutilities::gen_air_components(ic_tc_auto_ljung_nolog_est,
                                   ic_tc_auto_ljung_nolog_decomp,
                                   this_xtype = this_xtype,
                                   this_log = FALSE,
                                   this_stde = TRUE)

# convert the components into a list of tis time series objects

ic_tc_auto_ljung_nolog_comp_tis <-
   lapply(ic_tc_auto_ljung_nolog_comp, function(x)
      try(tis::tis(x, start = ic_this_start, tif = "wsaturday")))
```

Appendix B

```
proc ssm data=weekly1 breakpeaks plots=ao(normal) plots=maxshock;
* opt(tech=dbldog maxiter=200);
id date interval=week align=end;
*ods output ParameterEstimates=hyperparms
InformationCriteria=aic;
*** antilog disturbance variances;
 %if &slope=y %then %do;
  parms lvlevel lvslope lvseason lvirr;
 %end;
 %else %do;
  parms lvlevel lvseason lvirr;
 %end;
 vlevel  = exp(lvlevel);
 %if &slope=y %then %do;
  vslope  = exp(lvslope);
 %end;
  vseason = exp(lvseason);
 %if wm=y %then %do;
  vseasonm=exp(lvseasonm);
 %end;
  virr    = exp(lvirr);

*** define time-varying trend, seasonal, and irr components;
%if &slope=y %then %do;
  state level(1) type=ll(slopecov(d)=(vslope)) cov(d)=(vlevel)
   checkbreak;
  comp trend = level[1];
%end;
 %else %do;
  state level(1) type=rw cov(d)=(vlevel) checkbreak;
  comp trend = level[1];
 %end;

 state sinPart(&nfreq) type=RW cov(I)=(vseason);
 state cosPart(&nfreq) type=RW cov(I)=(vseason);
 comp sin = sinPart*(sin1-sin&nfreq);
 comp cos = cosPart*(cos1-cos&nfreq);

 state noise(1) type=wn cov(d)=(virr);
 comp irr = noise[1];

*** define model statement;
%if &ao=y and &tc=y and &ls=y %then %do;
 model &dv = trend sin cos irr
             &hm &aom &tcm &lsm
%end;

%if &ao=y and &tc= and &ls=y %then %do;
 model &dv = trend sin cos irr
             &hm &aom &lsm;
%end;

%if &ao=y and &tc= and &ls= %then %do;
 model &dv = trend sin cos irr &hm &aom;
%end;

*** use eval statements to sum up holiday and outlier components;
%if &ao=y %then %do;
 eval ao  = &aom_sum;
%end;
%if &tc=y %then %do;
 eval tc  = &tcm_sum;
%end;
%if &ls=y %then %do;
 eval ls  = &lsm_sum;
```

```
%end;
 *eval j4w = jul4wed;
* eval ltd = latethanks;
 eval hol = &hm_sum;
 eval sf  = sin+cos;

output out=smore ao(maxnum=10) press pdv gcv maxshock;
run;
quit;
```

References

Burman, J.P. (1980), “Seasonal Adjustment by Signal Extraction,” *Journal of the Royal Statistical Society Series A*, 143(3):321–337.

Cleveland, R., Cleveland, W.S., McRae, J., and Terpenning, I. (1990), “STL: A Seasonal-Trend Decomposition Procedure Based on Loess,” *Journal of Official Statistics*, v. 6, n. 1, pp. 3-73.

Cleveland, W.P. (1986), “Calendar Adjustment and Time Series,” Special Studies Paper 198, Division of Research and Statistics, Federal Reserve Board, Washington, DC.

Cleveland, W.P., Evans, T., and Scott, S. (2018), “Weekly Seasonal Adjustment: A Locally-weighted Regression Approach,” *Eurostat Handbook of Seasonal Adjustment*, Eurostat: Luxembourg.

De Jong, P. (1989), “The Diffuse Kalman Filter,” *Annals of Statistics*, v. 19, pp. 1073-1083.

Evans, T., and Sverchkov, M. (2016), “Variance Estimation for Weekly Seasonally Adjusted National UI Claims Series,” in *Proceedings of the Joint Statistical Meetings*.

Gómez, V., and Maravall, A. (1997), Programs TRAMO and SEATS: Instructions for the User (Beta Version: June 1997). Banco de España, Servicio de Estudios, DT 9628.

Gómez, V. and A. Maravall, A. (2001), “Seasonal Adjustment and Signal Extraction in Economic Time Series” in D. Pena, G. C. Tiao, and R. S. Tsay, editors, *A Course in Time Series Analysis*. J. Wiley and Sons, New York, NY.

Grudkowska, S. (2017), “JDemetra+ User Guide Version 2.2,” National Bank of Poland.

Harvey, A. (1989), *Forecasting, Structural Time Series Models and the Kalman Filter*, Cambridge, U.K.: Cambridge University Press.

Harvey, A., and Koopman, S.J. (1993), “Forecasting Hourly Electricity Demand Using Time-Varying Splines,” *Journal of the American Statistical Association*, v. 88, n. 424, pp. 1228-1236.

Harvey, A., Koopman, S.J., and Riani, M. (1997), "The Modeling and Seasonal Adjustment of Weekly Observations," *Journal of Business & Economic Statistics*, v. 15, n. 3, pp. 354-368.

Laidray, D., Palate, J., Mazzi, G.L., and Proietti, T. (2018), "Seasonal Adjustment of Daily and Weekly Data," *Eurostat Handbook of Seasonal Adjustment*, Eurostat: Luxembourg.

McElroy, T., and Livsey, J. (2020), "*Ecce Signum*: An R Software Package for Analyzing Multivariate Time Series," draft paper.

Maravall, A. (1985), "On Structural Time Series Models and the Characterization of Components," Journal of Business Statistics, v. 3, no. 4, pp. 350-355.

Monsell, B. (2020), "Using R in Seasonal Adjustment of Official Statistics," in JSM Proceedings, Business and Economics Section," Alexandria, VA: American Statistical Association.

Monsell, B. (2021), "Time Series Responses to the Covid-19 Pandemic at BLS for Monthly and Weekly Series," in JSM Proceedings, Business and Economics Section," Alexandria, VA: American Statistical Association.

Palate, J., rjdhighfreq: Routines for modelling and seasonal adjustment of high frequency series, Version 0.0.4. Bank of Belgium, Brussels, Belgium, August 2020 https://github.com/palatej/rjdhighfreq.

Pierce, D.A., Grupe, M.R., and Cleveland, W.P. (1984), "Seasonal Adjustment of the Weekly Monetary Aggregates: A Model-Based Approach," *Journal of Business & Economic Statistics*, v. 2, n. 3, pp. 260-270.

SAS Institute Inc. (2020), SAS/ETS® 15.2 User's Guide, Cary, NC: SAS Institute Inc. Available online at https://go.documentation.sas.com/api/docsets/etsug/15.2/content/ssm.pdf?locale=en#nameddest=etsug_ssm_toc.

Selukar, R. (2011), "State Space Modeling using SAS," *Journal of Statistical Software*, v. 41, issue 12. Available online at https://www.jstatsoft.org/article/view/v041i12.

U.S. Census Bureau Time Series Staff (2021), *X13ARIMA-SEATS Reference Manual* (version 1.1), Washington, DC: Authors. Available online at https://www2.census.gov/software/x-13arima-seats/x13as/unix-linux/documentation/docx13ashtml.pdf.